Knowledge-Based Reasoning in a Unified Feature Modeling Scheme

G. Chen[1], Y.-S. Ma[2], G. Thimm[3] and S.-H. Tang[4]

[1]Nanyang Technological University, pg02198079@ntu.edu.sg
[2]Nanyang Technological University, mysma@ntu.edu.sg
[3]Nanyang Technological University, mgeorg@ntu.edu.sg
[4]Nanyang Technological University, pg02104852@ntu.edu.sg

ABSTRACT

Feature-based modeling is an accepted approach to include high-level geometric information in product models as well as to facilitate a parameterized and constraint-based product development process. Moreover, features are suitable as an intermediate layer between a product's knowledge model and its geometry model for effective and efficient information management. To achieve this, traditional feature technology must be extended to align with the approach of knowledge-based reasoning. In this paper, based on a previously proposed unified feature modeling scheme, feature definitions are extended to support knowledge-based reasoning. In addition, a communication mechanism between the knowledge-based system and the feature model is established. The methods to embed and use knowledge for information consistency control are described.

Keywords: Unified feature; Knowledge-based reasoning; Feature-based modeling

1. INTRODUCTION

Historically, computer-aided tools, such as CAD, CAPP and CAM systems are developed to support the corresponding product lifecycle phases. Product geometry takes up a significant position in these systems. Feature technology is mainly used to provide high-level geometric representations to facilitate parameterized and constraint-based product geometric design process. However, traditional CAD systems usually assume that a designer has already finished the conceptual design as well as the concept-to-geometry mapping [17]. Therefore, only geometric modeling functions are provided in these CAD systems. Due to this limitation, some information, such as the design intent and the process patterns, is lost during the design process.

Explicitly embedding knowledge-based reasoning processes within the traditional engineering systems can significantly enhance the systems' reusability, scalability and flexibility. Knowledge-oriented techniques can support more complex tasks with natural human-oriented intelligent processes. They are more acceptable by human-beings than data-oriented techniques. They can also represent and deal with inaccurate and incomplete information as well as rule-of-thumbs, which are usually difficult to be described mathematically.

Due to the different natures between the knowledge information entities and the geometric entities, to interface the knowledge-based reasoning processes with the geometric modeling processes, an intermediate information layer is necessary. Feature-based technology can provide such a bridge. However, the current feature modeling technology has to be extended in two directions. First, feature definitions need to be extended to support knowledge-based reasoning processes. The purpose of this extension are checking and maintaining feature validity in the view of design intent. In other words, each feature model should be consistent to its specific knowledge bases. Second, the communication mechanism between the knowledge bases and the feature models must be established.

This paper is intended to address the key issues for enabling knowledge-based reasoning in a previously proposed unified feature modeling scheme. These issues include the alignment between the knowledge bases and the feature models, knowledge-based constraint-driven product modeling, knowledge updating and extraction throughout the unified feature modeling processes.

This paper is organized as follows: the next section reviews past research. In section 3, a three-part information model is proposed. In the fourth section, the information flows during the conceptual design and the detailed design phases are analyzed and then used to refine the feature definition from the viewpoint of knowledge-based reasoning. Relations between the knowledge bases, feature and geometry models are described in the fifth section. An implementation of these concepts is discussed briefly in the sixth section. Finally, conclusions are given.

2. LITERATURE REVIEW

Earlier feature definitions were mainly used to describe product geometry or as patterns for application-specific feature recognition [20]. Experience shows that features can be used as higher-level information entities going beyond pure geometry by grouping information entities together. For instance, the feature definition given in [8] includes feature attributes, topological relations and geometric constraints, which are ultimately based on a feature's geometric entities. These attributes, relations or constraints provide interfaces to higher level reasoning processes.

Brunetti *et al.* [2] proposed an entity-relation description of structure of a feature-based model and its features. A feature entity relation graph (FERG) is used in this context to describe intrinsic and extrinsic geometric relations between feature entities. Vieira [22] used constraints to specify and maintain feature semantics and validities. For this purpose, feature constraints were extended to include non-geometric semantic constraints, which represent the design intent of features. Relationships between features and higher level knowledge-based reasoning processes were explored. Mandorli *et al.* [12] proposed a self-validation feature concept to maintain feature validity. Bidarra and Bronsvoort [1] highlighted the importance of defining and maintaining feature semantics for a true feature modeling system. Regli and Pratt [18] discussed the influences of non-geometric feature interactions on feature validity. They all highlighted that the design intent of features is important in product design processes. However, in these publications, only results of the knowledge-based reasoning processes were implicitly represented via geometric or algebraic constraints.

Many other researchers focused on the conceptual design phase. Welch and Dixon [23] defined a function-behavior-embodiment representation scheme to map product functions to physical structures. Umeda *et al.* [21] proposed to use physical features to relate functions to behaviors and states. Pahl and Beitz [14] proposed a function-working principle-structured architecture to represent the conceptual design process of a product. A common limitation of these approaches is the absence of clear connections between the conceptual design and the detailed design.

Recently, some researchers made valuable explorations on connecting conceptual design systems to detailed geometric design systems. Ranta *et al.* [17] used intermediate structural entities and relations to link conceptual design and detailed design. Although no implementation detail about this intermediate stage was given, they proposed several important ideas for the integration of conceptual design and detailed design, such as the existence of generic ontologies between these two phases, common-function-based association among different geometric entities and the possible mapping from the abstract functional requirements to the physical geometric constraints. Brunetti and Golob [3] developed a representation scheme connecting the conceptual design and the detailed design phases. Xue and Yang [24] proposed an information structure to support concurrent design. Different product information entities, such as functions, geometry and machining processes, were included into a single product realization process graph. However, detailed descriptions of semantics of different application features, modification propagation and hence consistency control mechanisms were not mentioned. Zha *et al.* [25] analyzed the relationship between conceptual design and detailed design from an assembly-oriented viewpoint. Features were used as geometric entities as well as knowledge entities for assembly modeling, planning and evaluation processes. Roy and Bharadwaj [19] explored relations among product function, part behavior, geometry and specifications. They claimed spatial relationship alone can not represent product functions sufficiently. Energy transfer relations between part faces are also important to describe product functions and determine product specifications, such as fit types or surface finishes. All researchers seem to agree on the importance of including knowledge-based reasoning processes into product design systems.

Ma and Tong [10-11] revealed the importance of using and maintaining associative nature of features during a design process. An associative feature concept was proposed to model associated geometric feature entities via a generic feature class. It is highlighted that an ideal data structure of a feature definition must be flexible and self-contained. The consistency of relations has to be carried on while the different geometric representations are generated and managed. In the unified feature modeling scheme [4], this associative feature concept was extended to include non-geometric feature relations. The unified feature modeling scheme uses feature-based models as an intermediate information layer to connect knowledge bases to geometry models. In this paper, the refined feature definitions from the viewpoint of knowledge-based reasoning as well as the supporting connection mechanisms between the knowledge-based systems and the feature models are described. The basic idea here is to maintain product information validity and consistency through establishing and controlling more complete information relations.

3. THREE-PART FRAMEWORK FOR THE UNIFIED INFORMATION MODEL

Generally, in each product development phase, the product information model consists of three parts: a knowledge base, a feature model and a geometry model (Fig. 1.). The knowledge-based system may have several implementation schemes. For example, a rule-based expert system consists of three major parts: a knowledge base (a set of predefined rules and existing facts), working memory of loaded and newly derived facts; and an inference engine, which are used

to process rules and facts. Basic information entities used to represent knowledge include facts, actions and rules. Facts are data on which rules can reason. Actions are defined operations to be carried out to derive the new state. Rules represent human engineers' abstract reasoning processes. If a rule's antecedent pattern is matched, the rule is then activated (it might "fires"). Firing a rule may directly assert new facts or trigger the execution of some actions, which in turn can create new facts or activate other rules.
The geometry model represents the geometric and topological descriptions of a design. Because geometric modeling systems are inherently difficult to represent non-geometric information, the intermediate feature model is necessary to represent non-geometric information. With this layer, declarative knowledge bases and procedural geometry models are connected. In Fig. 1., arrows represent the information flow. Such information packages are based on a request and response manner following the standard XML protocols [13].

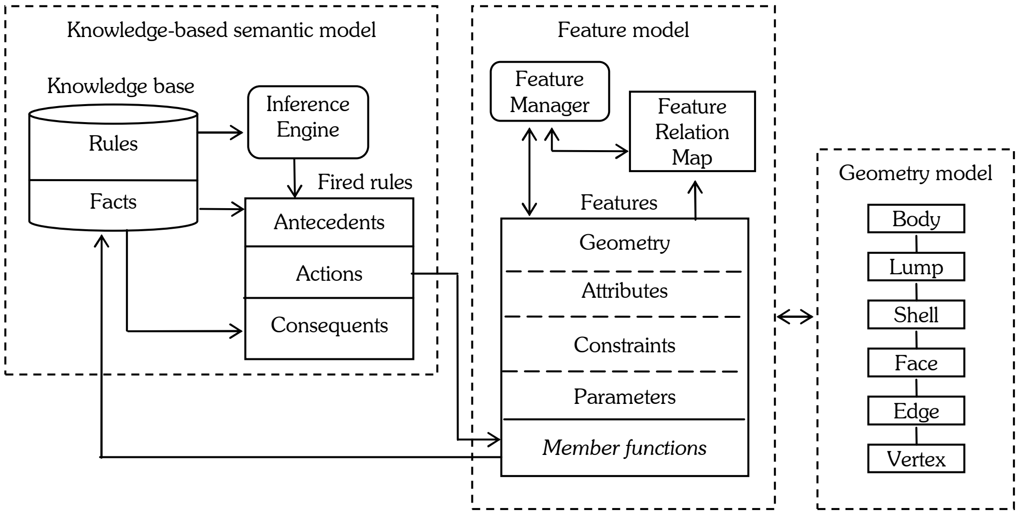

Fig. 1. Structure of a three-part product information model

The alignment between the knowledge bases and the feature models has two aspects, facts and actions. Facts are synchronized or extracted constantly through knowledge base agents with the necessary associations. Actions have to be mapped to feature member functions. It should be noted that features are defined in an object-oriented approach as classes. Therefore, communications between a feature instance and other entities is realized through its member functions (methods). This means the relations between a rule's actions and features' entities, such as attributes and constraints as well as the further down-stream connections to the geometric elements, are actually established using feature methods. Feature operations modify the existing facts or create new facts, or remove some old facts within the knowledge bases, therefore indirectly activate rules. To support this three-part information model, the feature definition has been refined from the viewpoint of knowledge-based reasoning. Based on the refined feature definition, relations between the knowledge-based system and the feature model are described in section 5.

4. UNIFIED FEATURE DEFINITIONS

A unified feature modeling scheme was proposed in [4] in which, feature unification and feature association were highlighted as two aspects of a feature-based integrated product information model. Feature unification provides a generic feature format and granularity for different phases while feature associations keep feature semantics and validities by maintaining feature relations. In the unified feature modeling scheme, features are defined as self-contained classes using object-oriented approach. In this section, the unified feature definition is described using the design process of an ejector pin in an ejection mechanism of a plastic injection mold assembly as an example. The information flows between the knowledge bases, feature models and geometry models during the conceptual design and the detailed design phases are described. The relevant entities of the respective feature classes are identified. Rationale of including knowledge-based reasoning process into a CAD system is derived along with the analysis process. It should be noted that Fig. 2. is just a sketch for illustration and hence some design details, such as draft angles, are not shown. In a molding process, the ejector components (ejector pins or stripper plate) form a part of the molding face. During the ejection process, the ejector assembly moves forward relative to the core plate, and the ejector pins (or stripper plate) remove the molding from the injection mold core. After the ejection, ejector pins are returned to their original positions by the closing mold cavity plate, which presses the return pins. Then the mold is ready for the next ejection. Fig. 2(a). shows the ejecting subsystem within a simple mold set when it is in the molding position. Fig. 2(b). shows the position after the ejection action.

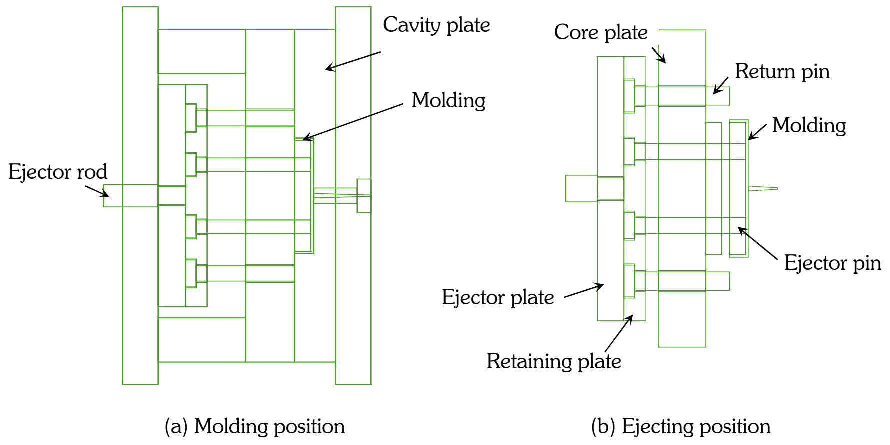

Fig. 2. The ejection mechanism in a plastic injection mold assembly

4.1 Features Defined in the Conceptual Design Phase

Major tasks in the conceptual design phase include function decomposition, function-physical structure mapping and the determination of critical dimensions/specifications. Knowledge-oriented techniques are used to simulate human engineers' reasoning processes. Mechanical functions and behaviors of a product are usually expressed as the interactions between pairs of critical faces in the form of relative positions, orientations, fit relations or relative motions as well as other critical dimensions/specifications. Here, a conceptual design feature model is used to represent and visualize these reasoning results. Each conceptual design feature instance corresponds to a relatively independent sub-function. Critical faces are features' geometric entities and other specifications are features' attributes or constraints. Different feature instances are connected through common functions.

For instance, in the ejection mechanism example, the main function of an ejection mechanism is the removal of the molded part from the injection mold [16]. This main function can be roughly decomposed into four sub-functions: 1) forming a part of the molding surface during the molding process; 2) ejecting the molding; 3) providing guidance during the ejection process; and 4) returning back to the original state after the ejection cycle. After mapping function-behavior-structure relations, critical product specifications are determined. In Fig. 3., four conceptual design features are illustrated, which are generated by using some predefined rules to represent these sub-functions respectively.

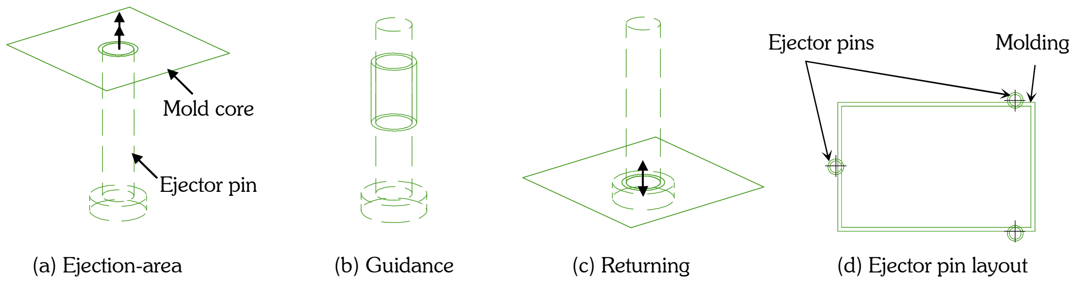

Fig. 3. Conceptual design features

One of the feature instances, the ejection-area's embedded alignment rule (refer to Fig. 3(a).) can be expressed as follows. Terms in the parentheses represent facts or actions. Words after the double forward slashes are comments:

Pin Conceptual Design Rule 1:

```
IF (sub-function is “forming a part of the molding surface during the molding process”)
   AND (ejection technique is “pin ejection”)
THEN (create an ejection-area feature instance) //alignment between pin head face and mold core face
        AND (pin head face and mold core face have the same surface definition)
        AND (pin head face and mold core face have the same orientation)
        AND (pin head face and mold core face have the same z-position)
        AND (pin head face and mold core face have the same surface finish specification)
```

According to the actions of this fired rule, an instance of a predefined conceptual design feature class, “ejection-area”, is generated (Fig. 3(a).). This new feature instance has two aligned faces. Feature attributes include surface definition, face orientation, position and surface finish. Some feature constraints are also initialized. In other words, relevant facts in knowledge space are mapped into the respective feature instances by manipulating their attributes, constraints as well as feature relations. Some relations between conceptual design feature instances are also generated, such as the orientation of the “guidance” feature instance has to comply with the face normal of the “returning” feature instance. These feature entities as well as relations between them are all generated from, correspond to and hence depend upon the actions of the fired rules. At the same time, the results of these actions will be extracted to update the existing facts or to create new facts in the knowledge base. Similarly, in the conceptual design phase, other necessary components are also initiated based on the functional features as shown in Fig. 3(b)., Fig. 3(c) and Fig. 3(d).

Based on the required functions and the predefined rules, it is very likely that the knowledge-based systems may generate many possible solutions, such as different function decomposition schemes, different function-structure mappings or a range of feasible values for critical dimensions/specifications. In these cases, the knowledge-based systems will ask designers to make a selection. These multiple solutions also provide opportunities for later design modifications and optimization.

4.2 Features Defined in the Detailed Design Phase

In this phase, the major tasks are finalizing geometric shape design and determining all necessary dimensions and specifications. Attributes and constraints specified in the conceptual design feature model should be kept and mapped into this new phase. Then these conceptual features are refined. Faces of previously generated conceptual design feature instances might be decomposed or recombined into faces of detailed design feature instances. In addition, supplementary structures and corresponding attributes/constraints need to be added. Knowledge-based reasoning in this phase is commonly used to deal with processes such as searching, matching and decision making over the constraints from other applications, e.g. assembly or manufacturing optimization. Such processes are difficult to be transformed into geometric or algebraic constraints. A typical rule for the selection of a D-shape ejector pin is shown:

Pin Detailed Design Rule 1:

```
IF (molding shape is “box type”) AND (wall thickness is “thin”) AND (feather edge is “none”)
   AND (molding size is “large”) AND (local shape is “none”)
THEN (D-shape pin for wall area) AND (Valve pin for central area)
```

Similar rules are developed to choose other types of ejector pins in an ejection mechanism, e.g. stepped pin, etc., according to the molding’s characteristics. By executing the actions of these fired rules, detailed product geometry can be created via initiating primitive form feature instances. Respective feature attributes and constraints are also specified. In Fig. 4., the D-shape pin is used for explanation. Assume this type of the ejector pin is chosen according to the characteristics of the molding (it is a thin-wall, box-type molding). Then the critical product specifications generated in the conceptual design phase are transformed into detailed design feature entities or feature relations. For example, the “ejection-area” conceptual feature is transformed into the alignment relation between the pin tip face and the local area on the mold plate as well as the molding surface in the molding position; the pin tip face shape is constrained with the mold plate surface and the core insert component edge (Fig. 4(b)., 4(c).). “Guidance” conceptual feature instance is transformed into the “sliding fit” relations among the pin component, core insert component and molding plate component (Fig. 4(b).). In this way, many constraints can be created according to the applicable but external and extendable rule sets. They are not necessarily predefined in the product model. However, the actions of such rules have to be matched with the corresponding feature methods predefined. Hence, some constraints can be applied dynamically upon the creation of feature instances. Therefore, such consistent match permits constraints to be added during feature generation and even in later modification processes. All the detailed constraints collectively ensure the realization of the required product functions. These kinds of constraints are difficult to be managed with "hard-coded" representation using geometric or algebraic constraints directly. Therefore, more flexibility and a higher level logic control are made possible.

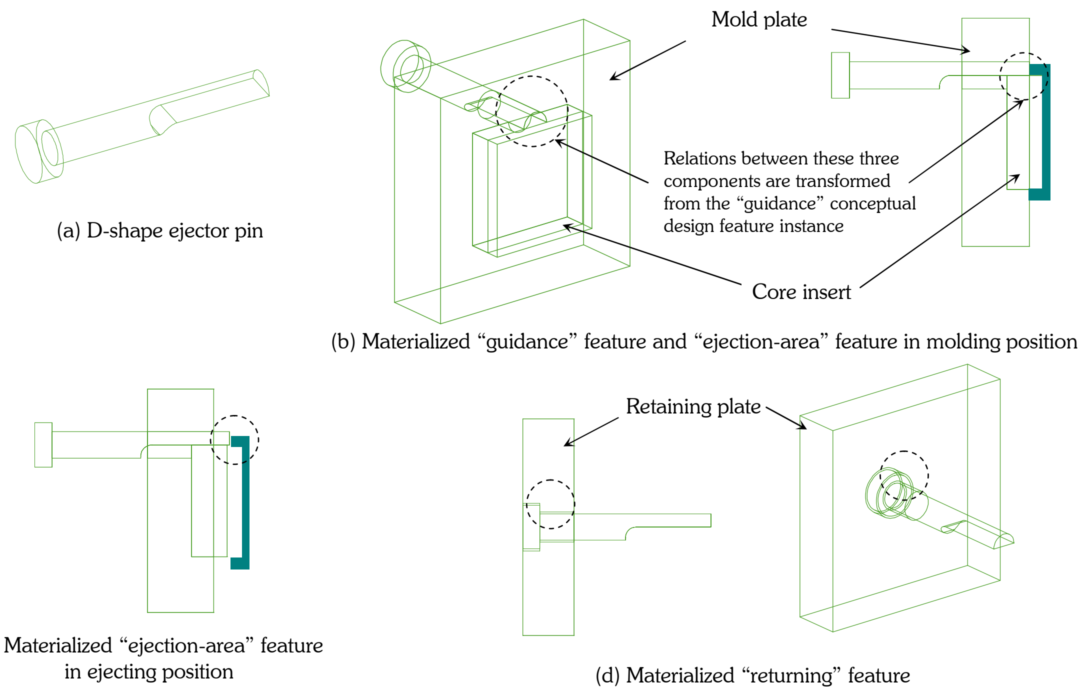
(a) D-shape ejector pin
Mold plate
Relations between these three components are transformed from the "guidance" conceptual design feature instance
Core insert
(b) Materialized "guidance" feature and "ejection-area" feature in molding position
(c) Materialized "ejection-area" feature in ejecting position
Retaining plate
(d) Materialized "returning" feature

Fig. 4. D-shape ejector pin in detailed design phase

4.3 Unified Feature

Based on the above analysis of information flows during the conceptual design and the detailed design phases, feature definitions used in the unified feature modeling scheme are discussed here. As an intermediate information layer, features must be interfaced upward to the knowledge base and downward to the geometry model. In the unified feature modeling scheme, a feature class consists of four types of member variables (geometric entities, attribute, constraint and parameter) and a set of member functions (such as geometry-creation, query, validity-check, etc.).

Geometric entities are a set of references. Each of them is associated with a unique geometric element in the geometry model. They do not interface with the knowledge bases directly but represent the interface to the product's geometry model.

Attribute entities describe feature's static properties, i.e. their current states. There are two types of attributes, geometry-relevant or geometry-irrelevant. Geometry-relevant attributes represent the evaluated results of intra-feature geometric properties or relations, such as diameter or evaluated results of feature constraints. Geometry-irrelevant attributes are usually meaningful in semantics among different product development phases. Constraint entities describe feature's dynamic properties. Constraints are used to delimit the domain of one or more features' attributes. There are two major types of constraints: geometric and algebraic. Geometric constraints specify relations between geometric entities, e.g., relative position or orientation, through reducing their degree of freedom. Algebraic constraints control the state of one or more attributes. Parameters are a special type of attributes, which are associated with constraints in a feature model, and the facts' state and design intent at the knowledge base level. Features also include reference variables which are owned or manipulated by other entities. Attribute and constraints together with parameters comprise a feature's interface to the knowledge bases. Part of the consequent facts of a fired rule is transformed into the corresponding attribute and/or constraints of feature instances. The type definitions of these resulted attributes and/or constraints, i.e. their classes, include lists of antecedents and/or consequents of rules that refer to the instances. The *Attribute* class also includes a list of constraints, which refer to the individual instances of it. Such unified feature definitions can be used generically across different phases, e.g. the conceptual design and the detailed design phases.

5. RELATIONS IN THE UNIFIED FEATURE MODEL

In order to use the knowledge-based systems to support feature validation and information consistency control process, relations between the knowledge bases and the feature models must be explicitly identified and then maintained during the feature modeling processes. In this section, relations in the unified feature model are analyzed.

5.1 Relations between the Knowledge Base and the Feature Model

The analysis in section 4 indicates that in different product development phases, modification of a feature model, such as choosing a feature type, initializing attributes or constraints, can be achieved by the means of knowledge-based reasoning, directly or indirectly. Feature types, attributes, and constraints depend on inferred information entities (actions, consequent facts) in the knowledge base (represented by the dashed arrows in Fig. 5.). On the other side, facts in the knowledge base can only be accurate when they are updated constantly from the feature entities. Generally, these dependencies represent relations between the knowledge base and the feature model. For instance, in the ejection mechanism example, the co-surface geometric constraint in the "ejection-area" feature instance depends on one of the asserted facts of Pin Conceptual Design Rule 1.

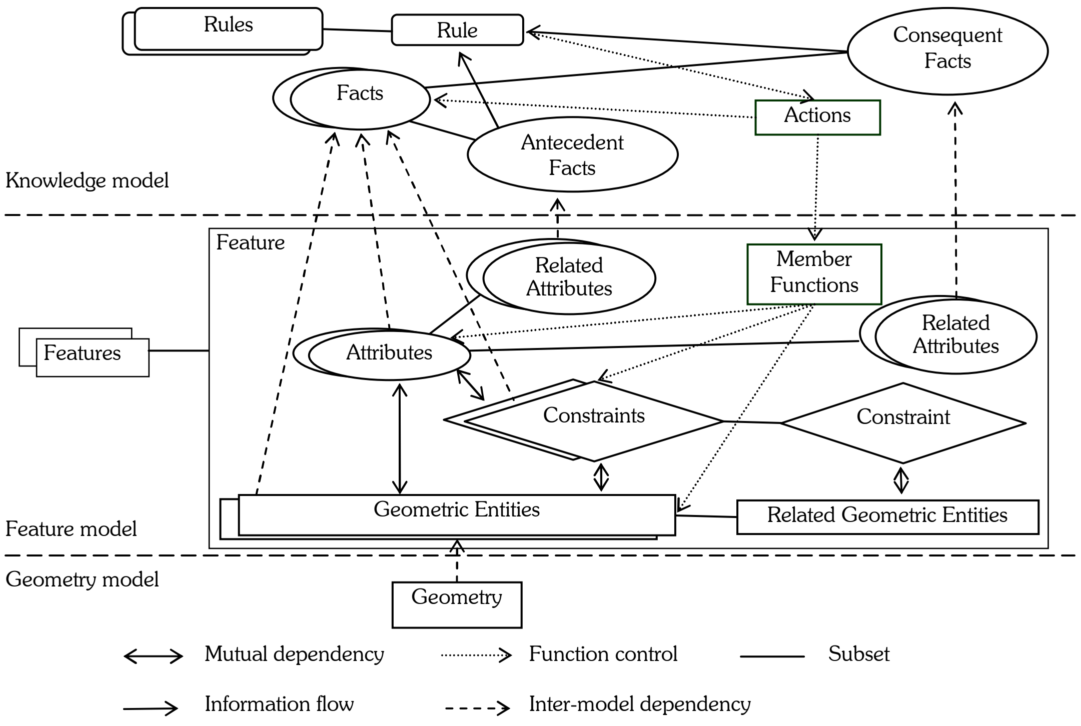

Fig. 5. Partial relations between the knowledge base and the feature model

Through forward chaining reasoning process, newly generated facts might trigger other rules to fire to assert new facts or invalidate previously fired rules. Feature operations might also change existing facts and hence trigger or invalidate rules. Logical constraints can be specified between the antecedent facts and the consequent facts or actions. That means the actions (feature operations) taken and the existence of the consequent facts depend on the existence of the antecedent facts. Whenever the antecedent facts of a rule are retracted, the corresponding actions, feature operations, consequent facts have to be retracted too. Therefore, those feature attributes or constraints derived directly from the knowledge-based reasoning processes, are associated with the respective rules. During a later feature modification, related rules need to be used to check the feature validity as well as to determine the modification's influence scope. In this way, feature validity check is extended to include knowledge-based reasoning. It is expected that based on the proposed integration, more flexibility in feature validity specifications can be achieved and more reasonable explanations can be provided to the users when a feature instance becomes invalid.

5.2 Intra- or Inter-Feature Relations

Intra-feature relations are classified into two categories. First, feature attributes are attached to their corresponding geometric entities (two-way solid arrows in Fig. 5.). These relations can be used to create/modify geometric elements in the forward direction or check feature attribute validity in the backward direction, i.e. whether values of features' attributes reflect the real situation of product's geometry model. Second, constraints control the geometry and attributes simultaneously (two-way solid arrows in the middle part of Fig. 5.). Relations also exist between features, which share the same geometric elements or are controlled by the same higher-level constraint, or together realize a product function, etc.

5.3 Relations between the Geometry Model and the Feature Model

Being an intermediate information layer, the feature model itself does not store geometric information. Each feature class has its geometry creation function. When a geometry creation function is invoked, feature geometry is created and inserted into the product geometry model. The generated geometric elements are associated to the feature instance's geometric entities. It is possible that several geometric entities of different feature instances associate to the same geometric element and vice versa.

Using these three-level relations, information generated during the knowledge-based reasoning processes are penetrated into and might be partially transformed into information entities in the feature model. Information, which is unsuitable or difficult to be represented in the feature model or the geometry model, is kept in the knowledge bases.

6. DEPENDENCY NETWORK AND ITS IMPLEMENTATION

In this section, implementations of the above identified relations in the unified feature modeling scheme are briefly described, object-oriented modeling techniques are used to define these relations while a justification-based truth maintenance system (JTMS) [6] is used to maintain dependency relations.

As mentioned in section 3, in each product development phase, the product information model consists of three parts: a knowledge base, a feature model and a geometry model. Due to this paper's focus, only the implementation of the knowledge base and the feature model are discussed here. Generally, in the knowledge space, the following major classes are defined: knowledgeBase, rule, pattern and fact. The instances of the knowledgeBase class maintain a list of predefined rule objects and existing fact objects. They are responsible for the forward-chaining reasoning, which includes matching rule patterns, selecting a rule to fire, asserting new facts or triggering actions, checking other facts and hence rules for their validities, asking the users for a selection, etc. The rule class records its antecedent patterns, consequent patterns and its current truth status (true, false or unknown). The pattern class has two subclasses, antecedent pattern and consequent pattern. The pattern class records its truth status and a list of the involved facts. It maintains a list of rule objects that refer to this pattern. The fact might be initial facts, user-provided facts or facts extracted from the feature space, such as feature attributes or constraints. The fact class also maintains a list of pattern objects that refer to this fact. When the value of a fact instance is changed (i.e. setValue function is executed), the truth status of all the referring patterns and hence the corresponding rules are checked.

In the feature space, the following major classes are defined: featureModel, feature, attribute and constraint. It is also a hierarchical model. In each level, it is an interaction network weaved by shared attributes and common constraints. Attribute and constraint objects can be initiated in different levels, such as assembly or feature constraints. Each attribute or constraint object maintains a list of other objects (pattern, assembly, feature, etc.) that refer to it. This list is used for change propagation and feature/rule validity check during design modifications. It must be noted here that in different domains, many other classes might be needed, such as the assembly and component classes in the detailed design phase.

Building dependency networks among design variables is regarded as an effective way to propagate design modifications and maintain information consistency. For example, Kusiak [7] proposed to use dependency analysis to resolve conflicted constraints. Park and Cutkosky [15] mentioned using dependencies at different abstraction levels to maintain information consistency. Ma *et al.* [9] also proposed a hierarchical model to represent geometric relations in a detailed design model. Eastman [5] proposed an algorithm of maintaining the integrity conditions between different interacted applications. Compared to these previous researches, the unified feature modeling scheme extends the scope of information consistency control to include the knowledge-based reasoning processes and hence provide more flexibility.

Design modification propagation in the proposed unified feature modeling scheme is roughly divided into two steps: propagation in the feature space and propagation in the knowledge space. Firstly, when the designer modifies (add, change or delete) a design variable, relevant variables (attributes or constraints) in the feature space are searched and checked for the corresponding modifications or validities. This kind of propagation is realized through the referring and binding relations between design variables. Referring means the associative relations between a feature and its used geometric elements via pointers. Binding relates a feature attribute to the corresponding feature geometric entities. At the end of these propagations, the newly given values or unsolved constraints are transferred back to the knowledge space via dependencies for further check of the related original design intents. Whenever a feature entity is initialized or later modified, its supporting justifications and corresponding antecedent patterns are recorded and associated with this feature entity. Rules or constraints can be used as supporting justifications of a feature entity. Facts, geometric entities or attributes can be used as antecedents. JTMS [6] algorithms are used to maintain consistencies.

This dependency network includes information entities of the knowledge, feature and geometry models. It is modified along with the different operations at different levels, e.g. feature creation or modification processes at the feature level. Inter-references can be searched in such cases that the attributes or constraints of feature instances depend on knowledge entities directly or indirectly via other feature entities. When a part is modified, this dependency network is traversed to find the affected feature entities. The validities of the affected feature entities are determined through checking whether their supporting justifications are still hold, i.e. whether the antecedent facts still persist and are valid. In the case of unsolved/conflict constraints in the feature space, other possible values in the knowledge space may be explored to find alternative design solutions. This demonstrates another advantage of including knowledge-based reasoning processes into a CAD system. General and domain-specific validity evaluation criteria and conflict resolution methods are developed to prevent infinite processing loops in large-scale problems.

The proposed general dependency network can be specialized into a specific domain, such as functional design or process planning. Keeping consistency between different product lifecycle phases is a future extension of the unified feature modeling scheme.

The implementation of a prototype system using Microsoft Visual C++ 6.0, Unigraphics V18.0, CLIPS V6.22 and MySQL 5.0.0 to demonstrate the proposed ideas is in progress. In the prototype system, an association manager module is developed to control the dependency network. Relevant product information entities as well as dependency relations are stored in a database.

7. CONCLUSION

In this paper, methods for incorporating the knowledge-based reasoning process into the product design systems are explored and proposed. Compared with the previous proposal [4], the unified feature definitions are refined for the purpose of managing dependency relations between the knowledge base and the feature model. The nature of such dependencies and the corresponding implementations are discussed. From this discussion, the authors conclude that the embedment of the knowledge-based reasoning capabilities can enhance the traditional CAD systems to a more human-oriented engineering process. For example, incomplete and inaccurate information can be dealt with. Function-related constraints, which are difficult to be represented by geometric or algebraic constraints, can be applied and managed. More design solutions might be explored for conflicts resolution or product lifecycle optimization. Intelligible explanations can be presented to the users when a modification resulted into unfulfilled constraints. In this way, information associations among the knowledge-base, feature and geometry models can be maintained and further used for feature validity check and information consistency control.

8. REFERENCES

[1] Bidarra, R. and Bronsvoort, W. F., Semantic feature modeling, *Computer-Aided Design*, Vol. 32, No. 3, 2000, pp 201-225.

[2] Brunetti, G., de Martino, T., Falcidieno, B. and Habinger, S., A relational model for interactive manipulation of form features based on algebraic geometry, *Proceedings of the third ACM symposium on Solid modeling and applications*, Salt Lake City, Utah, USA, 1995.

[3] Brunetti, G. and Golob, B., A feature-based approach towards an integrated product model including conceptual design information, *Computer-Aided Design*, Vol. 32, No. 14, 2000, pp 877-887.

[4] Chen, G., Ma, Y.-S., Thimm, G. and Tang, S.-H., Unified feature modeling scheme for the integration of CAD and CAx, *Computer-Aided Design & Applications*, Vol. 1, Nos. 1-4, *CAD'04*, 2004, pp 595-602.

[5] Eastman, C. M., Managing integrity in design information flows, *Computer-Aided Design*, Vol. 28, No. 6/7, 1996, pp 551-565.

[6] Forbus, K. D. and de Kleer, J., *Building problem solvers*, MIT Press, 1993.

[7] Kusiak, A., Dependency analysis in constraint negotiation, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 25, No. 9, 1995, pp 1301-1313.

[8] Laakko, T. and Mantyla, M., Feature modeling by incremental feature recognition, *Computer-Aided Design*, Vol. 25, No. 8, 1993, pp 479-492.

[9] Ma, W.-Y., Zhong, Y.-M., Tso, S.-K. and Zhou T.-X., A hierarchically structured and constraint-based data model for intuitive and precise solid modeling in a virtual reality environment, *Computer-Aided Design*, Vol. 36, No. 10, 2004, pp 903-928.

[10] Ma, Y.-S. and Tong, T., Associative feature modeling for concurrent engineering integration, *Computers in Industry*, Vol. 51, No. 1, 2003, pp 51-71.

[11] Ma, Y.-S. and Tong, T., An object-oriented design tool for associative cooling channels in plastic-injection moulds, *International Journal of Advanced Manufacturing Technology*, Vol. 23, No. 1-2, 2004, pp 79-86.

[12] Mandorli, F., Cugini, U., Otto, H. E. and Kimura, F., Modeling with self validation features, *Proceedings of the fourth ACM Symposium on Solid Modeling and Applications*, Atlanta, Georgia, USA, 1997.

[13] Muschamp, P., An introduction to web services, *BT Technology Journal*, Vol. 22, No. 1, 2004, pp 9-18.

[14] Pahl, G. and Beitz, W., *Engineering design: a systematic approach*, second edition, London, Springer, 1996.

[15] Park, H. and Cutkosky, M. R., Framework for modeling dependencies in collaborative engineering processes, *Research in Engineering Design*, Vol. 11, No. 2, 1999, pp 84-102.

[16] Pye, R. G. W., *Injection mould design: a textbook for the novice and a design manual for the thermoplastics industry*, fourth edition, Longman Scientific & Technical, 1989.

[17] Ranta, M., Mantyla, M., Umeda, Y. and Tomiyama, T., Integration of functional and feature-based product modelling – the IMS/GNOSIS experience, *Computer-Aided Design*, Vol. 28, No. 5, 1996, pp 371-381.

[18] Regli, W. C. and Pratt, M. J., What are feature interactions? *Proceedings of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference*, Irvine, California, USA, 1996.

[19] Roy, U. and Bharadwaj, B., Design with part behaviors: behavior model, representation and applications, *Computer-Aided Design*, Vol. 34, No. 9, 2002, pp 613-636.

[20] Shah, J. J. and Mantyla, M., *Parametric and feature-based CAD/CAM: concepts, techniques, and applications*, John Wiley & Sons, Inc., 1995.

[21] Umeda, Y., Ishii, M., Yoshioka, M., Shimomura, Y. and Tomiyama, T., Supporting conceptual design based on the function-behavior-state modeler, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 10, No. 4, 1996, pp 275-288.

[22] Vieira, A. S., Consistency management in feature-based parametric design, *Proceedings of the ASME 1995 Design Engineering Technical Conferences*, Volume 2, Boston, Massachusetts, USA, 1995.

[23] Welch, R. V. and Dixon, J. R., Representing function, behavior and structure during conceptual design, *Design Theory and Methodology – DTM'92*, ASME, New York, USA, 1992.

[24] Xue, D. and Yang, H., A concurrent engineering-oriented design database representation model, *Computer-Aided Design*, Vol. 36, No. 10, 2004, pp 947-965.

[25] Zha, X.-F., Du, H.-J. and Qiu, J.-H., Knowledge-based approach and system for assembly oriented design, Part I: the approach, *Engineering Applications of Artificial Intelligence*, Vol. 14, No. 1, 2001, pp 61-75.